MANAGEMENT OF CONFIGURATION DATA FOR MULTI-DEVICE SYSTEMS

FIELD OF THE INVENTION

[0001] The present invention generally relates to managing configuration data for systems having a plurality of configurable devices.

BACKGROUND

[0002] Some systems contain many devices that are configurable. One type of configurable device is a programmable logic device (PLD). Examples of programmable logic devices include field programmable gate arrays (FPGAs), complex programmable logic devices (CPLDs), programmable array logic (PAL), and programmable logic arrays (PLAs). As systems are developed with increasing numbers of configurable devices, management of the configuration data for these devices becomes increasingly problematic. Challenges may arise from having different design teams involved in developing the configuration data for a system, as well as from different versions of configuration data created for custom applications and different releases created to improve the system or fix problems.

[0003] For a complex system, different design teams may be responsible for developing the design for different functional areas of the system. In some instances, the design teams may be using different servers for data storage, and the directory structure used in managing the configuration data may vary by design team. Thus, the configuration data for a system may be spread across numerous storage domains and not uniformly organized.

[0004] Different versions of configuration data generated during the product development cycle and maintenance activities also complicate management of configuration data. One or more devices in a system may have different configuration data from one release to the next. Each new release adds to the number of configuration data files to be managed. Thus, there may be numerous files of configuration data to deal with for a single system.

[0005] For maintenance activities, identifying the correct configuration data for a system may also be problematic. In systems made with programmable logic devices (PLDs), such as field programmable gate arrays (FPGAs) or complex programmable logic devices (CPLDs), the same physical hardware components may take on different personalities based on the configuration. Thus, one system may appear to be physically the same as another system, but the configuration of the systems may make them slightly or even very different from one another. Identifying a system and then selecting the correct configuration data may present various challenges to technicians and users.

[0006] The present invention may address one or more of the above issues.

SUMMARY OF THE INVENTION

[0007] The various embodiments of the invention relate to managing configuration data for a system. In one embodiment, a chain description data set is generated to specify an order in a configuration chain of configurable devices in the system and identify configuration data sets associated with the configurable devices. A system identifier value may be generated and associated with the chain description data set. An archive may be generated including the configuration data sets, chain description data set, and system identifier value.

[0008] In other embodiments, the stored system identifier value may be used to verify whether the archived configuration data is suitable for a target system. Still other embodiments preserve a directory hierarchy of configuration data files in the archive for portability and ease of restoration.

[0009] It will be appreciated that various other embodiments are set forth in the Detailed Description and Claims which follow.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Various aspects and advantages of the invention will become apparent upon review of the following detailed description and upon reference to the drawings in which:

[0011] FIG. 1 is a block diagram of an arrangement for identifying an electronic system that includes a plurality of electronic devices;

[0012] FIG. 2 is a flowchart of an example process for identifying an electronic system in accordance with various embodiments of the invention;

[0013] FIGs. 3A, 3B, 3C, and 3D illustrate example embodiments of electronic devices from which a system identification code may be determined;

[0014] FIG. 4 illustrates an electronic system in which configuration data from each of the devices in the system may be used to identify the system;

[0015] FIG. 5A illustrates an example process for collecting the configuration data associated with configuring an electronic system having a plurality of programmable logic devices in accordance with various embodiments of the invention;

[0016] FIG. 5B is a data flow diagram that illustrates an example arrangement of configuration data sets assembled in an archive in accordance with various embodiments of the invention; and

[0017] FIG. 6 is a flowchart of an example process for configuring an electronic system using configuration data collected in accordance with various embodiments of the invention.

DETAILED DESCRIPTION

[0018] The various embodiments of the invention assist in particularly identifying electronic systems. In one

embodiment, identifier codes are read from the electronic devices that comprise the system. The system identifier is determined based on the values of all the identifier codes read from the system. Depending on implementation requirements, a device may be manufactured with a fixed identifier code, a user may configure the device with a user-specific identifier code, the identifier code may be determined from pin states of the device, or various combinations of these approaches may be used to identify a system.

[0019] In various other embodiments, the configuration data associated with an electronic system may be managed in association with the system's unique identifier. Archiving configuration data files in a selected directory structure and associating the system identifier with the archive enables portability of the configuration data and reduces the chances of improperly configuring a system.

[0020] FIG. 1 is a block diagram of an arrangement 100 for identifying an electronic system 102 that includes a plurality of electronic devices 104-1 - 104-*n*. In an example environment, a software tool 106 may be used to learn the identity of an electronic system 102. The tool is coupled to the system via system interface 108, which may include the software drivers and hardware interface needed to communicate with the devices 104-1 - 104-*n*.

[0021] The general functions provided by tool 106 may support testing, configuration, or even application functions directed to end users of the system. The ability to obtain the identity of a system may be useful for activities such as verifying compatibility between the tool and the system, verifying license rights, or learning a release level of the system. It will be appreciated that various commercially available or proprietary tools and system interfaces may be adapted for use with the embodiments of the present invention.

[0022] In various embodiments, the system identifier may be generated from selected device-identifying signals read from all the devices in the system. That is, the collection of signals read from all the devices may be used to derive a system identifier code. Alternatively, the system identifier code may be generated from a selected subset (fewer than all) of the devices in the system. For example, the identifier may be generated from each odd or even numbered device in a scan chain. Different combinations of devices may be used to identify the system according to implementation requirements.

[0023] In one embodiment, the boundary-scan architecture defined in IEEE standard 1149.1 may be used to support identifying an electronic system. It will be understood that "JTAG" is sometimes used to refer to the 1149.1 standard and similar boundary-scan standards. With JTAG, each of devices 104-1 - 104-*n* has a boundary-scan register into which scan data is serially input to the device or output from the device. The devices are connected in a scan chain with the scan output pin (TDO) of one device feeding the scan input pin (TDI) of the next device in the chain. The system interface 108 is coupled to each of the devices 104-1 - 104-*n* with control lines 110 (e.g., TMS, TCK) for controlling the input and output of scan data to the devices. Lines 112 and 114 illustrate the serial input and serial output data lines. In some embodiments, a memory, for example a non-volatile memory, may be coupled to the device and store an identifier for the device. In such embodiments, the stored identifier may also be used to derive the system identifier code.

[0024] It will be appreciated that other types of interfaces, standard or proprietary, may be used to implement an interface to electronic devices for learning device identification codes.

[0025] FIG. 2 is a flowchart of an example process for identifying an electronic system in accordance with various embodiments of the invention. The states of selected signals are read from selected devices of the system (step 200). The selected signals are those signals chosen to identify the device according to implementation requirements. As previously explained, different implementations may use all of the devices on the system or various subsets of devices of the system for learning the system identifier code.

[0026] The particular signals that are read from a device to learn a device identification code may vary from implementation to implementation. It will be appreciated that the approaches to learning device identifier codes described herein may be used alone or in various combinations to meet implementation objectives.

[0027] In one approach, the contents of the boundary-scan IDCODE register may be used to identify a device (see FIG. 3A). The contents of the IDCODE register are generally set by the device manufacturer and specify information such as the identification code and revision number for the device. In some embodiments, the IDCODE register is non-programmable, or has limited programmability (e.g., a one-time programmable fuse).

[0028] In another approach, because the IDCODE register may not be specific enough to identify a system for some implementations, the boundary-scan USERCODE register may be used either alone or in combination with the IDCODE register to more particularly identify a device and thereby a system (see FIG. 3B). The USERCODE register is programmable via the boundary-scan TDI interface, which allows a device to be particularly identified according to user requirements. It will be appreciated that the "user" may be, for example, the system vendor, system maintainer, or end user. Examples of

values that may be stored in the USERCODE register include a checksum of configuration data from a PLD or a value derived from the configuration data and the position of the device on the scan chain.

[0029] In another approach, the boundary-scan register (BSR) may be used to determine the identity of a device (see FIG. 3C). In particular, the boundary-scan EXTEST and SAMPLE (or SAMPLE/PRELOAD) instructions may be used to access the boundary-scan register. The EXTEST instruction causes the TDI and TDO pins to be connected to the boundary-scan register. The pin states of the device may be sampled and new values shifted into the boundary-scan register. Then, the new values may be applied to the pins of the device.

[0030] The SAMPLE instruction also causes the TDI and TDO pins to be connected to the BSR. However, the device is left in its normal functional mode. During this instruction, the boundary-scan register may be accessed by a data scan operation to sample the functional data input to and output from the device. The SAMPLE instruction may also be used to load data into the boundary-scan register prior to loading an EXTEST instruction.

[0031] In another approach, the configuration data in a programmable logic device may be used to identify the device (see FIG. 4). The configuration data from a device may be read back and a checksum value computed from the configuration data. The checksum value identifies the device and may be used in combination with the checksum values from other devices to particularly identify the system.

[0032] The system identifier code is generated from the device identifier codes (step 202). There may be many suitable functions for generating a system identifier code from a set of device identifier codes. For example, in one implementation, a concatenation of the device identifier codes may be used as the

system identifier code. In another implementation, a combination of the positions of the devices in the scan chain along with the device identifier codes may be used to generate a system identifier code. It will be appreciated that various other transformations of the device identifier codes, and position if desired, may be alternatively used.

[0033] The system identifier code may then be saved for subsequent use (step 204). For example, the system identifier code may be useful for licensing authorization, confirming an upgrade level, or identifying system capabilities.

[0034] In another embodiment, the identification code may be stored in a non-volatile memory and later used to detect unauthorized reconfiguration of the system. For example, a controller may periodically read device identifier codes and generate a new system identifier code, and then compare the new system identifier code to the stored code. If the new system identifier code does not match the stored system identifier code, the system may be halted or an electronic message may be sent to a party interested in the mismatch.

[0035] FIGs. 3A, 3B, 3C, and 3D illustrate example embodiments of electronic devices 302, 304, 306, and 308, respectively, from which a system identification code may be determined. Each of the example devices is implemented with logic that complies with the boundary-scan architecture.

[0036] FIG. 3A illustrates a device 302 in which the IDCODE register 309 is connected to the TDI and TDO pins of the device 302. The connection is made in response to input of an IDCODE instruction to the boundary-scan logic (not shown), and the register's contents are shifted out on the TDO pin.

[0037] FIG. 3B illustrates a device 304 in which the USERCODE register 310 may be used alone or in combination with the IDCODE register 312. The USERCODE register 310 is connected to the TDI and TDO pins in response to input of a

USERCODE instruction to the boundary-scan logic, and the register's contents are shifted out on the TDO pin.

[0038] FIG. 3C illustrates a device 306 in which the boundary-scan register 314 is connected to the TDI and TDO pins. The boundary-scan SAMPLE instruction may be used to obtain the states of device pins in the boundary-scan register and shift out the contents via the TDO pin. In using the SAMPLE instruction, the device is left in its normal functional mode. The boundary-scan register can be accessed by a data scan operation to take a sample of the functional data entering and leaving the device while selected pins of the device are held at a selected signal levels, as illustrated by signal lines 316, 318, and 320. Thus, by setting the values at signal lines 316, 318, and 320 appropriately, the device may be identified.

[0039] FIG. 3D illustrates a device 308 in which the boundary-scan register 314 is connected to the TDI and TDO pins, and a boundary-scan EXTEST instruction may be used to obtain a fixed value from the boundary-scan register. An output pin of device 308 is driven to logic level 1 and provided on line 322 to a multiplexer 323 to perform system identification. When the signal on line 322 is logic level 1 the multiplexer selects the identification signals on lines 324, and when the signal is logic level 0 the multiplexer selects the normal input signals on lines 326. The signals on lines 324 are held at a selected signal levels to perform the system identification. The multiplexer or equivalent functionality may be implemented external to the device or possibly as part of the internal device logic.

[0040] FIG. 4 illustrates an electronic system 402 in which configuration data from each of the devices 404-1 - 404-*n* in the system may be used to identify the system. Certain PLDs, such as various FPGAs from Xilinx, Inc., may be connected in serial for loading and reading back configuration data. The

devices have pins for input of configuration bits. For example an input pin of device 404-1 is connected to line 406. Similarly, the devices have output pins for output of configuration bits. For example, a configuration output pin of device 404-*n* is connected to line 408. The configuration bits and boundary-scan bits may use the same pins for input of data, as illustrated. In other implementations, the configuration bits and scan bits may be input on separate pins.

[0041] Each of devices 404-1 - 404-*n* is configured with a respective configuration data set 414-1 - 414-*n*. Each configuration data set programs the associated device to implement desired functionality. These same configuration data sets may be read back from the devices (using conventional tools) and collectively used to generate a system identifier code by which the system 402 may be particularly identified, as explained above.

[0042] The identification of an electronic system may be used in managing configuration data for a system that includes multiple programmable logic devices, such as shown in FIG. 4. In a system having multiple PLDs, the number of configuration data sets to be managed grows with each system revision or update. The number of configuration data sets to be managed may be compounded in situations where configuration data sets must be managed for multiple systems.

[0043] FIG. 5A illustrates an example process for collecting the configuration data associated with configuring an electronic system having a plurality of programmable logic devices in accordance with various embodiments of the invention. FIG. 5B is a data flow diagram that illustrates an example arrangement of configuration data sets assembled in an archive in accordance with various embodiments of the invention. Various elements of FIG. 5B, as well as elements in

FIG. 4, may be referenced in describing the process flow of FIG. 5A.

[0044] The process may commence with identifying the configuration data sets required for configuration of a system (step 502). For example, each of configuration data sets 414-1 - 414-*n*, may be stored in a networked or local file system 504 (FIG. 5B), and organized in a hierarchical directory structure. For example in file system 504 (FIG. 5B), configuration data set1 and set2 are stored in directory dir3, which is a subdirectory of dir2, which is a subdirectory of dir1. Identifying the configuration data sets may involve a user specifying (user control 506) to a management tool 508 the names of the files in which the configuration data sets are stored and the directory paths to the files. Alternatively, one or more databases may be used to store the configuration data sets, and part of the identification may involve specifying keys associated with the configuration data sets.

[0045] In one embodiment, a separate directory structure may be created containing unique configuration directories for each of the configuration data sets (step 512). The separate directories are created to avoid having to change the name of a file of configuration which might be the same as the name of another file of configuration data. Each file of configuration data is then copied to the configuration directory created for that file (step 514). The separate directory structure is illustrated by block 516 (FIG. 5B).

[0046] The process also creates and stores data that is later used to generate a configuration bitstream for the system (step 518). Configuration bitstream refers to the correctly ordered stream of data that is input to the system for configuration of the various configurable devices. For example, the configuration bitstream for system 402 (FIG. 4) includes the configuration data sets 414-1 - 414-*n* in the

proper order for shifting the configuration data sets into the respective devices. Chain description data specifies information relating to the various configurable devices, such as the device order in the scan chain and the configuration data set (or "file") associated with each device. In an example embodiment, a user may specify to the management tool 508 the device order and configuration data sets associated with the devices. In an alternative embodiment, the management tool 508 may read device identifier codes from the system 520 for determining the order of devices and require the user to match the device identifiers to the proper configuration data sets.

[0047] A system identifier code is generated and stored with the chain description data (step 522). The system identifier code may be generated as described in the discussion of FIGs. 1-4.

[0048] An archive file 524 (FIG. 5B) is created using the new directory structure of configuration data 516 (FIG. 5B), the chain description data, and the system identifier code (step 526). In archiving the configuration data sets, the new directory structure 516 is preserved in the archive so that when the data is extracted, the files are extracted into the same new directory structure and the configuration bitstream 530 (FIG. 5B) may be easily assembled. The archive file may be created using known operating system utilities. Archiving the configuration data sets and associated information allows the complete data set for configuration a system to be easily moved from one site or system to another.

[0049] The archive file may be optionally compressed (step 528). For example, if the archive is a Zip file the data is automatically compressed. For other types of archives, additional utilities may be used for compressing the data. The archive file may also be optionally encrypted for additional

data security. When encryption is used, the data will be decrypted only when an appropriate matching decryption key is provided.

[0050] FIG. 6 is a flowchart of an example process for configuring an electronic system using configuration data collected in accordance with various embodiments of the invention. The process of FIG. 6 assumes that the configuration data has been collected using the process of FIG. 5A. The configuration process may be implemented as functions provided by management tool 508 (FIG. 5B).

[0051] If the archive file selected for use in configuring a system is compressed, the archive is decompressed (step 602). The data in the archive is extracted (step 604). As indicated above, the directory structure of the files of configuration data is preserved in extracting the files from the archive so that the files used to construct the configuration bitstream may be easily located. The location to which the data is extracted may be selected by the user.

[0052] The process then determines the identifier of the target system (step 606). The target system is the electronic system being considered for configuration with the data from the archive. The target system identifier may be determined using the approaches described in the discussion of FIGs. 1-4.

[0053] The system identifier extracted from the archive is read (step 608) and compared to the target system identifier (decision step 610). If the identifiers are not equal, the configuration process is halted (step 612). The mismatch of identifiers indicates that the extracted configuration data is inappropriate for the target system. This may save the user from unknowingly creating problems.

[0054] If the identifiers match, a configuration bitstream is created from the extracted files of configuration data sets (step 614). The extracted chain description data is used to

appropriately order the configuration data sets in the configuration bitstream. Once the configuration bitstream is assembled, the chain of devices in the system may be configured (step 616).

[0055] It is also possible to use this mechanism when applying upgrades to systems. In this scenario, the target system is already configured. The upgrade might only be applied if the system identification value of the target system as currently configured is in an allowed list for the upgrade. This prevents upgrades that may be incompatible with the existing system from being applied.

[0056] Those skilled in the art will appreciate that various alternative computing arrangements would be suitable for hosting the processes of the different embodiments of the present invention. In addition, the processes may be provided via a variety of computer-readable media or delivery channels such as magnetic or optical disks or tapes, electronic storage devices, or as application services over a network.

[0057] The present invention is believed to be applicable to a variety of systems for managing configuration data for electronic systems and has been found to be particularly applicable and beneficial in uniquely identifying electronic systems from the collection of device identifiers and associating unique system identifiers with the configuration data of the system. Other aspects and embodiments of the present invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. It is intended that the specification and illustrated embodiments be considered as examples only, with a true scope and spirit of the invention being indicated by the following claims.